




# Ordering and Making Pizza for 120 Million

## Designing and Building the Next Office

© 2000 Microsoft Corporation  
All rights reserved. Microsoft, the Microsoft Dynamics logo, and the Microsoft Dynamics logo are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

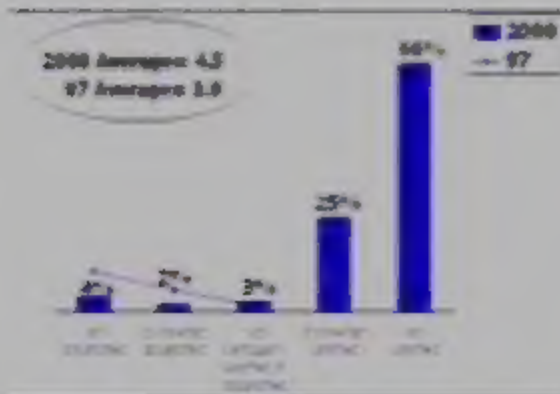


# Agenda

- Designing a New Office
  - Office customer challenges
  - Understanding our customers
  - Using customer feedback, understanding in design
- Building Office
  - How we're organized
  - The development cycle
  - The code
  - The life of an Office developer
  - What we do well
  - What we don't do so well


# Challenges – The Pizza Problem...

- 120 million relatively happy and loyal customers
  - Why should I get the next version?




## Highlights

- 2/3 of users give Office 2000 a rating of 5 out of 5
- 90% are at least somewhat satisfied with Office 2000



## Challenges – The Pizza Problem...

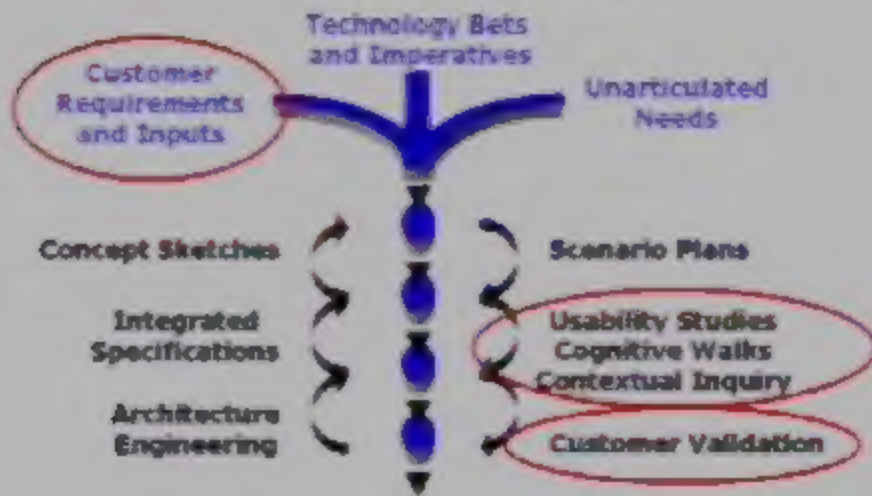
- Wide spectrum of target customers
  - From end-users to administrators
- Want new features but no new learning
  - How do I migrate the users and the current learning
- Must be innovative but fully compatible
  - New features but no surprises (file formats, macros, solutions)
- Designed to be flexible but lockable
  - Fully customizable and configurable – but lockable if I so wish
- Fully loaded but not bloated
  - My 20% of the features are most important
  - The other 80% should not get in my way



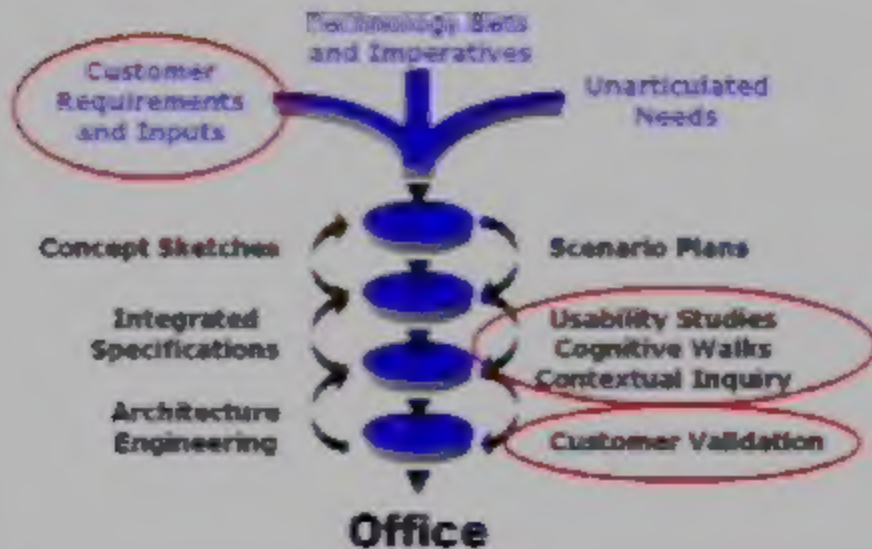
# Understanding the customer better


- Instrumented versions
  - Log everything done by users
  - Some "duhs", some surprises
- Usability studies
  - Get to see and feel the customer's pain
- Customer visits
  - Everyone does them
  - Watch users work
- Advisory Council
  - From 15 hand-picked companies
  - Meet in Redmond before and during dev cycle

# Designing for innovation



# Designing for innovation





# How We're Organized

- Roles:

- Core:

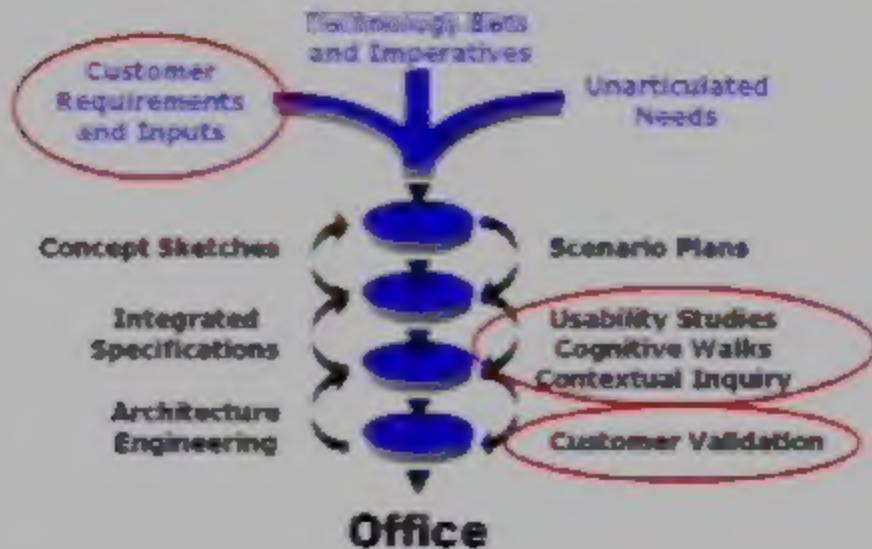
- Dev, Test, Prog. Mgmt., User Ed., Marketing


- Others:

- Product Planners
    - Designers
    - Usability Specialists
    - PSS
    - Localizers



# Designing for innovation





# How We're Organized

- Roles:
  - Core:
    - Dev, Test, Prog. Mgmt., User Ed., Marketing
  - Others:
    - Product Planners
    - Designers
    - Usability Specialists
    - PSS
    - Localizers

# How We're Organized

- Groups

- Application Teams

- End User Test in Teams
    - Develop application features

- Office Shared Feature Teams

- End User Test in Teams
    - Develop shared features
    - Responsible for shared code - integration

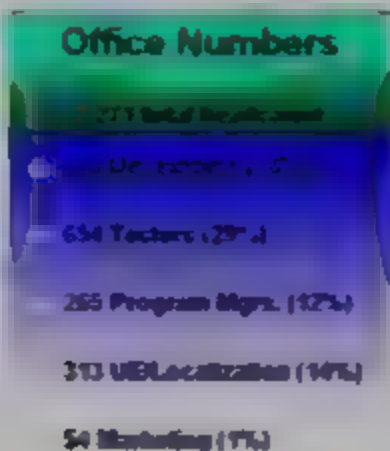
- Office Core Code

- Maintain the Office Core Code

- UI/UX

- Develop UI/UX

# How We're Organized



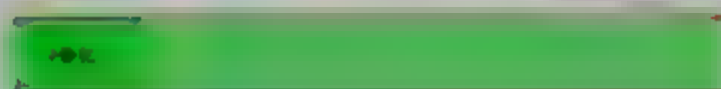
# Development Timeline



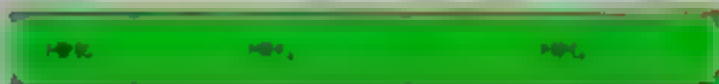
# Development Timeline



# Development Timeline

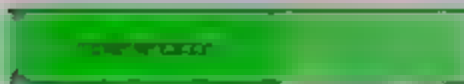
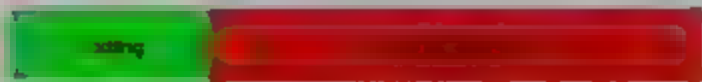


# Development Timeline

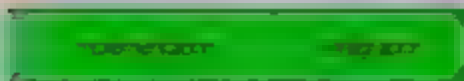
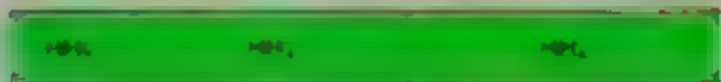




# Development Timeline



# Development Timeline



# Development Timeline



# Development Timeline



# Development Timeline



# Development Timeline



# Development Timeline



# Development Timeline

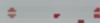




# Development Timeline



# Development Timeline



## The life of an Office developer

- Features are assigned

# The life of an Office developer

- Features are assigned
- Designs are done
- Design reviews
- Features are scheduled
- Schedule is rolled up and balanced

## The life of an Office developer

# The life of an Office developer

- Implementation starts
- Different for Office shared developers - App for all users
- The end state
  - Implementation finished
  - Implementation to app store & dev
  - Update needed
    - App store development
    - Github repo
    - App store deployment
  - Final app review

## The life of an Office developer

# The life of an Office developer

- Life centered around the bug fix
- Needs analysis
  - Total count
  - Bug size
  - Priority, eg beta bugs
- Group stage • bug in at some point
- UET stage • bug in a little later



# About the Code

- All apps written in C
  - Excel Word
- or in C++
  - MSC Access PowerPoint Outlook

# About the Code

How much code is there?

# About the Code

How much code is there?

## Office 2000

Office	1,173,700
Word	1,121,500
Excel	1,456,000
Outlook	1,408,500
Access	1,173,300
PowerPoint	750,000
<b>Total</b>	<b>7,705,230</b>

# About the Code

How much code is there?

## Office 2000

File	3,179,765
Word	3,021,500
Excel	1,456,111
Outlook	1,408,250
Access	1,179,298
PowerPoint	709,463

**Total 7,705,230**

## Office 97

File	2,511,220
Word	2,176,500
Excel	1,111,296
Outlook	645,450
Access	362,908
PowerPoint	420,131

**Total 4,056,270**

# About the Code

- Word in Office 2000
  - 24,817 words
  - 145 sentences making changes
  - Tracked 331,423 lines of code
  - 30% of changes to 1% of sentences
  - 100% growth in 7 years

# What we do well

## for all disciplines

- Product vision
- Setup from day one
- Teams organized by feature areas not apps
  - where separate people
- Unified test harness
  - Example: Results recording
- Rigorous peer inspections
- Knowledge Management
  - Have too late to put feedback
  - Have information for everyone
  - Have a place to put things

http://office10

Microsoft Office 10

Welcome to Office 10

# What we do well

## for developers

- Single unified build system

  - Easy for us, we don't have to manage

  - the "real" images & files, the "development" images

  - have pre-installed images for the build machines

- Regular complete builds

  - Dependencies

  - Cache & tests

  - Including external code & tools &

- Up-to-date, 2013

  - Full features set available

  - Almost 100%



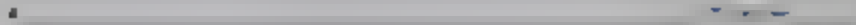
## AssertTags

## Office 10 D.W. Debug AssertTag Table

## AssertTags (cont.)



# OfficeWatson



# What we do well for developers

- Easy to use performance tools
  - Dedicated performance team
  - Some of our own lightweight tools
- Coding Scheduling methodologies
  - Bug age limit during in-house coding
  - Saved-in debug time to schedule
  - Debugging as TDD too
    - Thread list changes
    - TDD framework
    - Thread execution technique

# Check-in tool

2. Add a new check-in



New Check-in


1. Add a new check-in

1. Add a new check-in

1. Add a new check-in


1. Add a new check-in

1. Add a new check-in




## Things we're not good at

- Maximizing coding time
  - Coding to overhead ratio too high
  - Need more robust builds/tests
    - Too much time blocked
    - Too many "random" failures today
    - Some tools are too fragile
    - The "environment" is too fragile
  - Build break resolution/turn-around timeliness



## Things we're not good at


- How do we reduce bug fixing time?
  - Improve checkin code quality
    - Verification tools
    - Better developer discipline
    - Pre-checkin test depth/targeting
    - Staging and sub-projects
  - Reduce the number of "spec issue" bugs
    - One analysis shows this is 40% of all bugs



## Things we're not good at

- Reducing bug fixing time (cont.)
  - Need better way of prioritizing bugs
    - Gather more data internally & from betas
  - Performance roller coaster
    - Too much "break it now, fix it later"





## Things we're not good at

- Schedule predictability
  - Better data
  - Better model of how things really are
  - Schedule *everything*